Analysis Specification On

Suicide Prevention

Suman Khadka

00162827

Computing Project

Level 5 Diploma in Computing

Softwarica College of IT and E-Commerce

Kathmandu, Nepal

April 23, 2017

Contents

[2. Analysis specification 3](#_Toc480788896)

[2.1. Introduction 3](#_Toc480788897)

[2.2. Requirements Specification 4](#_Toc480788898)

[2.2.1 Operating Requirement 6](#_Toc480788899)

[2.2.2 External User Interface Requirements 6](#_Toc480788900)

[2.2.3 Functional Requirements 6](#_Toc480788901)

[2.2.4 Non-Functional Requirements 10](#_Toc480788902)

[2.3. Prioritization 12](#_Toc480788903)

[2.4. Use Case Diagram 14](#_Toc480788904)

[2.5. Architecture 17](#_Toc480788905)

[2.5.1. System Architecture 17](#_Toc480788906)

[2.5.2. Initial Class Diagram 19](#_Toc480788907)

[2.6. Conclusion 19](#_Toc480788908)

# **2. Analysis specification**

## **2.1. Introduction**

Analysis is the most difficult part of software development life cycle. It is the foundation of system development in waterfall approach. This stage helps to analyze the system both in hardware and software perspective. Most of the clients do not know what they want from the system. So, analysis carried out by project team can help them understand client’s requirements clearly. It helps to document the requirements properly in **Requirements Specification.**

Why to carry out Analysis?

* Analysis helps to understand the problem domain of a system.
* Functional and Non-Functional requirements can be documented.
* Stakeholders of the system are identified.
* Better communication between the stakeholders is obtained resulting in quality software development.
* Ambiguous requirements are clarified in analysis specification.
* The architecture of the possible system can be worked out through class diagram.

Proper analysis is the milestone to success in system development. Analysis specification consists of following stages:

1. [Requirements Specification](#_Requirements)

The requirements of the system are gathered and organized in systematic approach in requirements specification.

1. [Prioritization](#_Prioritization):

The requirements of system are prioritized in the order to develop effective system. MoSCoW prioritization is used to carry out prioritization of each requirements.

1. [Use Case Diagram](#_Use-Case_Diagram)

Use Case Diagram helps to identify the users in the system. The system, the user, and the activities performed by the users are identified in the form of diagram in Use Case.

1. [System Architecture](#_System_Architecture)

It helps to determine a conceptual model that helps to identify the structure, behavior of the system to be developed. The architecture or framework stated at the analysis phase may or may not be completely accurate and complete.

The requirements specification is discussed in [Section 2.2](#_2.2._Requirements). The prioritization is discussed in [Section 2.3.](#_2.3._Prioritization) and Use Case Diagram is illustrated in [Section 2.4](#_2.4._Use_Case) and System Architecture in [Section 2.5.](#_2.5._System_Architecture)

## **2.2. Requirements Specification**

Projects have certain requirements that need to be fulfilled. These needs are acquainted with the help of client as the product is developed for client themselves. Requirements ease the overall development of a system. Requirements are often misunderstood in system development. This can be a serious issue which leads to project failure. So, the requirements are captured and gathered in systematic approach. Software Requirements Specification(SRS) is prepared for well documented description of the system to be developed. It helps to identify the functional and non-functional requirements of a system. IEEE standard is used to develop SRS document. SRS helps to establish an agreement between Client and Developer. Once the SRS document is accepted, the project is further succeeded to other steps.

Why carry out Requirements Specification?

* It helps to resolve ambiguity in system development.
* It helps developers what to develop.
* It helps to create a mutual understanding between client and developers.
* It helps to produce well documented requirements specification which is easy to read for non-technical personals.

For the project, several methods were used to acknowledge as much requirements as possible. Some of them were as follows:

1. Questionnaires:

Questionnaires is the way of collecting information in which participants are supposed to answer the given questions. Several questions were prepared for the questionnaires. The participants were from the field who are constantly working in order to prevent suicide.

1. Online Survey:

Online survey was carried out to analyze user appreciation and requirements on the system. The survey is still live at: <https://freeonlinesurveys.com/s/FdtdQVEl> . The survey’s website is shown below:

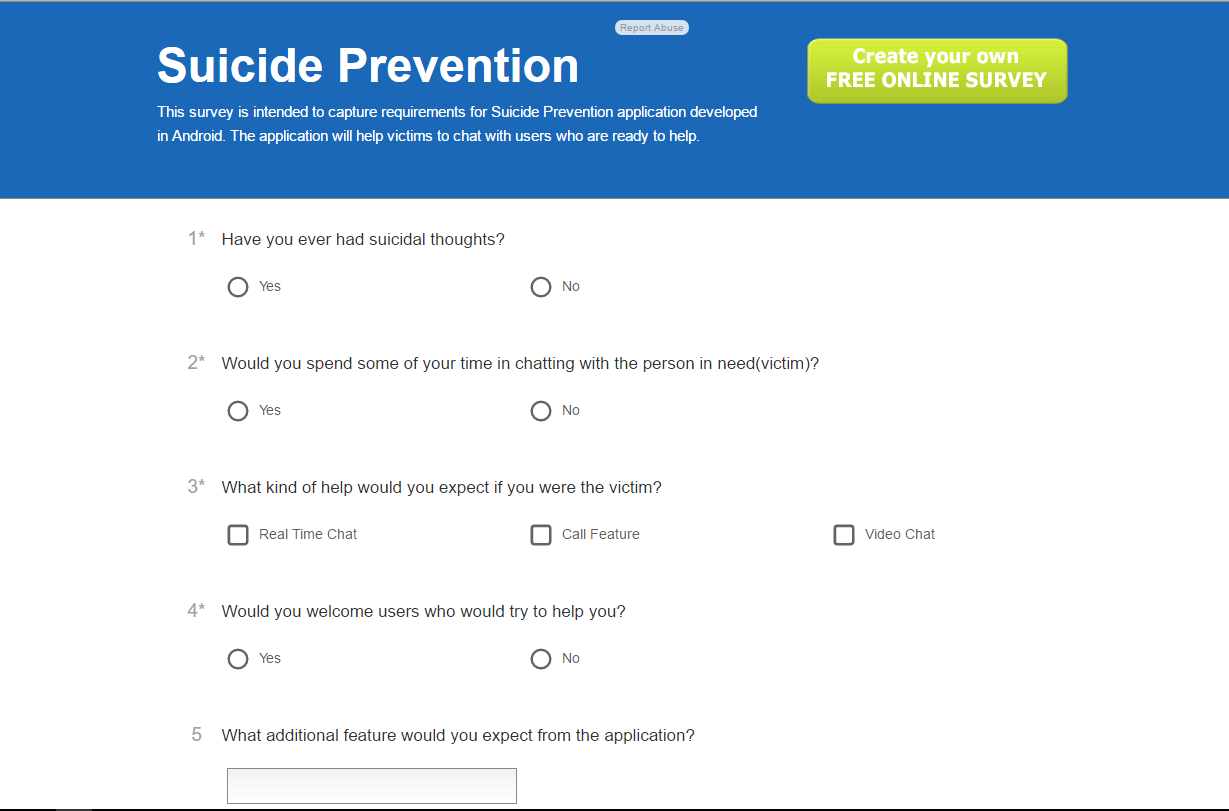




Figure : Survey Website

Most of the participants from the survey showed positive response towards the development of application. With the help of survey, more user appreciation was obtained and additional requirements were gathered.

1. Research:

Research also plays vital role in requirement gathering. Research was carried out on the respective subject to understand the system and analyze requirements.

All the requirements are systematically categorized into respective terms. The categories are:

### 2.2.1 Operating Requirement

The operating environment of the application is going to be in android operating system. The user must have android system in order to use the application. Android with 4.3 Jellybean and above is more preferred for better user experience. The application won’t run on devices with another OS’s beside android OS.

### 2.2.2 External User Interface Requirements

The user expects the application to be easy to use and more user friendly. The user interface makes huge impact on users. The user should be able to process input easily and be able to see output clearly. The color, layouts and fonts of the system will be chosen in such a way that the application will be more accessible to users.

### 2.2.3 Functional Requirements

* + - 1. *User Class: Victim/Helper*

**ID:** FR1

**Title:** Application Installation

**Description:** Users must be able to install the application in their android devices.

**Rational:** In order for a user to install application.

**Dependencies:** N/A

**ID:** FR2

**Title:** User Registration

**Description:** Users have to register themselves into the system.

**Rational:** To verify user’s identification and login credentials.

**Dependencies:** FR1

**ID:** FR3

**Title:** User Login

**Description:** Users have to login to the system.

**Rational:** To allow users to use the application securely and maintain privacy.

**Dependencies:** FR2

**ID:** FR4

**Title:** Manage Profile

**Description:** Users can manage their profile after logging in.

**Rational:** To make the user’s description more precise and valid.

**Dependencies:** FR3

**ID:** FR5

**Title:** Search Available Helpers

**Description:** Victims can get list of available helpers.

**Rational:** To allow victims to choose appropriate helpers.

**Dependencies:** FR3

**ID:** FR6

**Title:** Search Available Victims

**Description:** Helpers can get list of available victims.

**Rational:** To allow helpers to help victims.

**Dependencies:** FR3

**ID:** FR7

**Title:** Chat

**Description:** Users can choose to chat with selected helper/victim.

**Rational:** To provide user help through messaging.

**Dependencies:** FR3

**ID:** FR8

**Title:** Commend Helper

**Description:** Victim can commend helpers if they wish to. Commend is term which shows gratitude towards their action.

**Rational:** To make helpers’ profile more valuable and increase commend points in helper’s perspective.

**Dependencies:** FR3

**ID:** FR9

**Title:** Create New Post

**Description:** Users can create a new post in which they can share their thoughts.

**Rational:** To allow users to share their thoughts to public.

**Dependencies:** FR3

**ID:** FR10

**Title:** View Posts

**Description:** Users can view posts from other users.

**Rational:** To allow users to feel others’ thoughts.

**Dependencies:** FR3

**ID:** FR11

**Title:** Delete Post

**Description:** Users can delete their post if they wish to.

**Rational:** To add delete functionality in posts.

**Dependencies:** FR8

**ID:**  FR12

**Title:** Call other users

**Description:** Users can call other users. Victims can call helpers and vice-versa.

**Rational:** To add call functionality.

**Dependencies:** FR3

**ID:** FR13

**Title:** Report

**Description:** Users who misbehave can be reported.

**Rational:** To allow users to make reports to impolite users.

**Dependencies:** FR3

* + - 1. *User Class: Moderator*

**ID:** FR14

**Title:** Create Account

**Description:** Moderators can create account.

**Rational:** To get moderator’s sign in credentials.

**Dependencies: N/A**

**ID:** FR15

**Title:** Login

**Description:** Moderators can log into the system.

**Rational:** To access the system’s other functionality.

**Dependencies:** FR13

**ID:** FR16

**Title:** See List of Reports

**Description:** Moderators get the list of reports from users.

**Rational:** To analyze the report and take appropriate action.

**Dependencies:** FR13

**ID:**  FR17

**Title:** Ban User

**Description:** Moderator can ban user who is offensive or impolite.

**Rational:** To balance the system with positive vibes.

**Dependencies:** FR14

### 2.2.4 Non-Functional Requirements

**ID:**  NR1

**Title:** Performance

**Description:** Higher performance is expected. The output should be fast and quick. Less resources should be used.

**Rational:** To obtain user satisfaction from fast performance.

**Dependencies:** N/A

**ID:**  NR2

**Title:** Accuracy

**Description:** The data in the system must be accurate and precise.

**Rational:** To improve accuracy.

**Dependencies:** N/A

**ID:**  NR3

**Title:** Modifiability

**Description:** The application must be modifiable in future releases.

**Rational:** To upgrade software more easily.

**Dependencies:** N/A

**ID:**  NR4

**Title:** Portability

**Description:** The system is expected to run on most of Android devices.

**Rational:** To target more users.

**Dependencies:** N/A

**ID:**  NR5

**Title:** Reliability

**Description:** Application must be reliable and should perform its functional requirements.

**Rational:** To allow users to use the system.

**Dependencies:** N/A

**ID:**  NR6

**Title:** Security

**Description:** The security of system should be maintained.

**Rational:** To maintain data privacy.

**Dependencies:** N/A

**ID:**  NR7

**Title:** Usability

**Description:** Users should be able to learn the application quickly and easily.

**Rational:** To target more users with basic knowledges.

**Dependencies:** N/A

**ID:**  NR8

**Title:** Availability

**Description:** Application must be available all the time.

**Rational:** To provide 24/7 support.

**Dependencies:** N/A

## **2.3. Prioritization**

Prioritization is essential to determine which requirements are important in software development. Prioritization helps to rank all the requirements relative to prioritization techniques. There are many techniques to prioritize requirements. MoSCoW prioritization is used in this project. MoSCoW focuses on requirements as Must Have, Should Have, Could Have, won’t have but Would Have. All the requirements are categorized and distinguished from each other. The ‘must have’ requirements must be in the system. The should have requirements should be in the system but are not vital. The could have requirements are not necessarily required in system. They are mostly optional. Won’t have requirements are not included in project for this release due to time or resource constraints.

Why Prioritizations?

* It helps to focus on necessary requirements
* It saves time and effort.
* It helps to use available resources carefully at right situation.
* User acceptance is achieved which is a road to success.

Victim/Helper:

|  |  |
| --- | --- |
| Requirements | MOSCOW |
| FR1. Application Installation | Must Have |
| FR2. User Registration | Must Have |
| FR3. User Login | Must Have |
| FR4. Manage Profile | Should Have |
| FR5. Search Available Helpers | Must Have |
| FR6. Search Available Victims | Must Have |
| FR7. Chat | Must Have |
| FR8. Commend Helper | Could Have |
| FR9. Create New Post | Must Have |
| FR10. View Posts | Must Have |
| FR11. Delete Post | Must Have |
| FR12. Call Other Users | Could Have |
| FR13. Report | Should Have |

Moderator:

|  |  |
| --- | --- |
| Requirements | MOSCOW |
| FR14. Create Account | Should Have |
| FR15. Login | Should Have |
| FR16. See List of Reports | Should Have |
| FR17. Ban User | Should Have |

Non-Functional Requirements:

|  |  |
| --- | --- |
| Requirements | MOSCOW |
| NR1. Performance | Should Have |
| NR2. Accuracy | Should Have |
| NR3. Modifiability | Should Have |
| NR4. Portability | Could Have |
| NR5. Reliability | Should Have |
| NR6. Security | Must Have |
| NR7. Usability | Must have |
| NR8. Availability | Must Have |

By prioritizing using MoSCoW, it is clear that some of the requirements are more important than others. So, requirements which are important are focused while developing software.

## **Use Case Diagram**

Use Case Diagram helps to clarify the system’s requirements and its users. Use Cases are generally prepared after analyzing the requirements itself. There are mainly actors, system, process, and relationship in the diagram.

* Actor: Actor is simply user who uses the system. There are many actors in system. In our system, there are three actors namely: Victim, Helper, and Moderator.
* Process: It is the activity performed by the actor in the system.
* Relationship: It helps to link actors and user processes.

Why Use Case Diagram?

* To gather requirements
* To get external and internal factors of system
* To get outside view of system.



Figure: Use Case Diagram of Helper

Use Case Diagram of Helper:

1. Helper can create account.
2. Helper can login into the system.
3. Helper can manage his/her profile after logging in.
4. Helper can search for victims who are online.
5. Helper can chat with victim.
6. Helper can view posts from other users
7. Helper can create new post.
8. Helper can delete his/her post.
9. Helper can call victim.
10. Helper can check availability of user.
11. Helper can report other users.

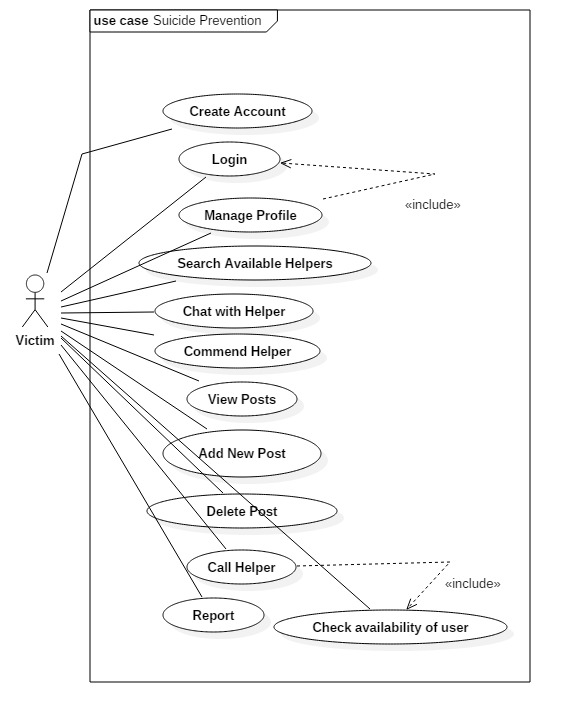


Figure: Use Case Diagram of Victim

Use Case Victim:

1. Victim can create account.
2. Victim can login into the system.
3. Victim can manage his/her profile after logging in.
4. Victim can search for helpers who are online.
5. Victim can chat with helper.
6. Victim can commend helper to show level of appreciation.
7. Victim can view posts from other users
8. Victim can create new post.
9. Victim can delete his/her post.
10. Victim can call victim.
11. Victim can check availability of user.
12. Victim can report other users.

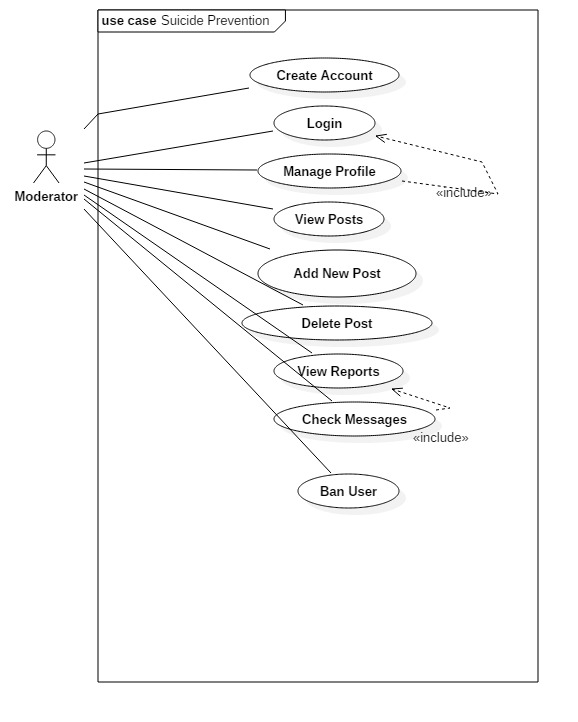


Figure: Use Case Diagram of Moderator

Use Case Moderator:

1. Moderator can create account.
2. Moderator can login into the system.
3. Moderator can manage his/her profile after logging in.
4. Moderator can search for helpers who are online.
5. Moderator can chat with helper.
6. Moderator can view posts from other users
7. Moderator can create new post.
8. Moderator can delete his/her post.
9. Moderator can view reports from other users.
10. Moderator can check messages of users who were reported.
11. Moderator can ban the offensive user.

## **Architecture**

An architecture specifies the overall structure of a system. It specifies the fundamental parts of a system that needs to be developed. There are several architectural patterns in computing project.

### 2.5.1. System Architecture

The application will be developed using android’s native system architecture. Android applications use pseudo-MVC architecture. In android, Activity classes are Controllers, Models are Business Logic, Views are XML layouts. So, the project will be following the same architecture. When an application follows well-defined architecture, there is greater chance of not failing the project. The project will be using pseudo-MVC architecture.

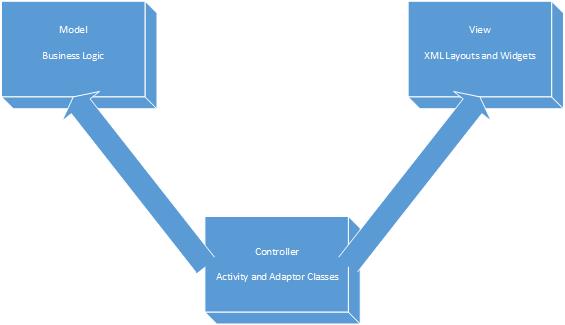


Fig: MVC in Android

The application will be using Firebase Database Server. So, the application will 2-tier application. User will connect directly to firebase server and retrieve information.

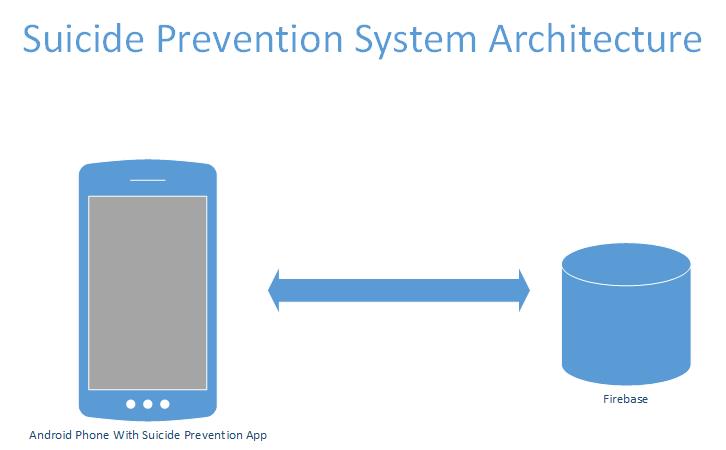


Figure 2: Firebase and Application

### 2.5.2. Initial Class Diagram

Class diagram represents the static time independent view of system. The initial class diagram shown below is produced using Natural Language Analysis. Classes identified below are at domain level.

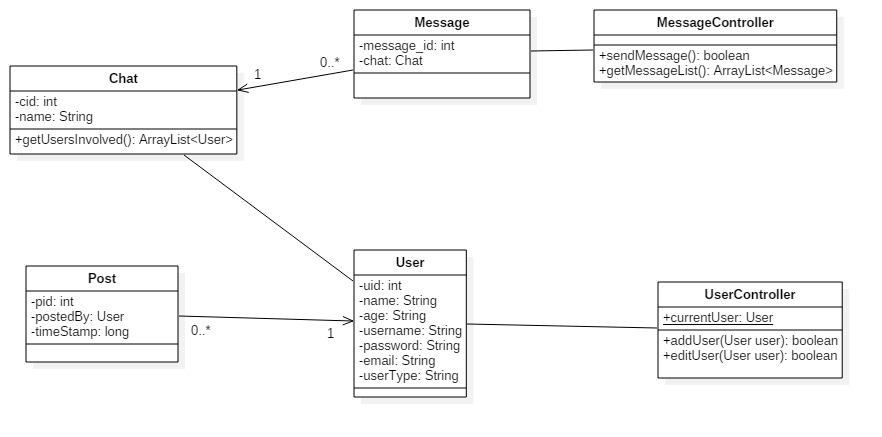


Figure: Initial Class Diagram.

Four models Chat, Post, User and Message are produced in class diagram. MessageController will be used as controller of Message Model. It will consist of functions to manipulate Message related requirements. Chat can contain zero or more messages but message can contain only one Chat. Similarly, User can post multiple times but one post cannot be posted by multiple users. User Sign up, user edit and many other features are controlled from UserController class.

In this way, Initial Class Diagram is completed. The diagram is not complete. So, it requires further classes and many other stuffs, which will be fulfilled in design specification.

## **Conclusion**

Analysis Specification was completed at the predicted date with all the requirements known. Now, the requirements and system architecture with initial diagram is known to us. The architecture of the system is also well defined which will help executing the requirements into system systematically. It will help us create design specification and ultimately complete our project successfully.